UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/728,441	12/01/2000	Wen-mei W. Hwu	042302 0269900	3226

27498 7590 07/17/2006

PILLSBURY WINTHROP SHAW PITTMAN LLP
P.O. BOX 10500
MCLEAN, VA 22102

EXAMINER	
LI, AIMEE J	

ART UNIT	PAPER NUMBER
2183	

DATE MAILED: 07/17/2006

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No.	Applicant(s)	
	09/728,441	HWU ET AL.	
	Examiner	Art Unit	
	Aimee J. Li	2183	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *01 May 2006*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-50,53 and 54* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☒ Claim(s) *32-34* is/are allowed.

6)☐ Claim(s) *1-31, 35-50, 53, and 54* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: _____.

DETAILED ACTION

1. Claims 1-50, 53, and new claim 54 have been considered. New claim 54 has been added as per Applicant's request. Claims 1, 9, 11, 13, 26, 32, 35, and 43 have been amended as per Applicant's request. Claim 52 has been cancelled as per Applicant's request.

Papers Submitted

2. It is hereby acknowledged that the following papers have been received and placed of record in the file: Amendment as received 01 May 2006.

Allowable Subject Matter

3. Claims 32-34 are allowed.

4. The following is an examiner's statement of reasons for allowance: Claim 32 recites

> ...a plurality of buffers respectively associated with the plurality of functional units, each buffer adapted to store the kernel set of loop instructions, and each buffer further adapted to store a different set of scheduling information associated with the respective functional unit and the stored instructions; and
>
> ...wherein during a first processor cycle, the control logic uses the sets of scheduling information to cause a portion of the prologue set of loop instructions to be issued to the respective functional units from the buffers, and wherein during a second processor cycle...the control logic uses the sets of scheduling information to cause a portion of kernel set of loop instructions to be issued to the respective functional units from the buffers, and wherein during a third processor cycle...the control logic uses the sets of scheduling information to cause a portion

of the epilogue set of loop instructions to be issued to the respective functional units form the buffers.

5. The combination of these elements, i.e. the plurality of buffers associated with the functional, each buffer storing the same set of kernel loop instructions and scheduling information for the respective functional unit, and executing the prologue, kernel, and epilogue loop instructions based upon the scheduling information on specific cycles.

6. Any comments considered necessary by applicant must be submitted no later than the payment of the issue fee and, to avoid processing delays, should preferably accompany the issue fee. Such submissions should be clearly labeled "Comments on Statement of Reasons for Allowance."

Claim Rejections - 35 USC § 103

7. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

8. Claims 1-4 and 8-18 are rejected under 35 U.S.C. 103(a) as being unpatentable over Mahalingaiah, U.S. Patent Number 5,989,865 (herein referred to as Mahalingaiah) in view of George, U.S. Patent Number 4,626,988 (herein referred to as George).

9. Referring to claim 1, Mahalingaiah has taught a processor comprising:

a. A functional unit adapted to execute an instruction issued to it from a dispatch stage (Mahalingaiah column 4, lines 32-43; column 11, lines 13-37; and Figure 1); and

b. A buffer in the dispatch stage further adapted to store scheduling information associated with each of the instructions (Mahalingaiah column 17, line 66 to column 18, line 16; and Figure 4). In regards to Mahalingaiah, the MROM Access stores all of the MROM instructions, which includes all loop instructions that includes the kernel instructions.

c. Wherein the plurality of the instructions comprise a kernel set of instructions from a loop body (Mahalingaiah column 2, lines 35-47; column 17, line 66 to column 18, line 16; and Figure 4). In regards to Mahalingaiah, all of the microcode loop instructions, including the kernel set of instructions in the loop, for a string MROM instruction are stored in the MROM Access.

d. Wherein the instructions are issued to the functional unit from the buffer in accordance with the stored scheduling information so as to cause the functional unit to execute a number of iterations of the body (Mahalingaiah column 17, line 66 to column 18, line 16; and Figure 4).

10. Mahalingaiah has not taught a buffer in the dispatch stage coupled to the functional unit adapted to receive from a fetch stage and store a plurality of the instructions before issue to the functional unit. George has taught a buffer in the dispatch stage coupled to the functional unit adapted to receive from a fetch stage and store a plurality of the instructions before issue to the functional unit (George Abstract; column 1, lines 26-47; and column 1, line 55 to column 2, line 68). A person of ordinary skill in the art at the time the invention was made, and as taught by George, would have recognized that the loop buffer eliminates redundancy, reduces storage contention, and providing efficient management for loop mode operations (George column 1,

lines 26-47 and column 2, lines 57-68). Therefore, it would have been obvious to a person of ordinary skill in the at the time the invention was made to incorporate the loop buffer of George in the device of Mahalingaiah to eliminate redundancy, reduce storage contention, and provide efficient management for loop mode operations.

11. Referring to claim 2, Mahalingaiah has taught a decode stage register coupled between the dispatch stage and the functional unit, the functional unit coupled to the decode stage register for executing the instruction issued to the decode stage register from the dispatch stage (Mahalingaiah column 4, lines 32-44; column 9, lines 6-16; and Figure 1).

12. Referring to claim 3, Mahalingaiah has taught control logic coupled to the buffer adapted to cause a certain one of the stored plurality of instructions to be issued to the functional unit in accordance with a loop iteration stage (Mahalingaiah column 17, line 66 to column 18, line 16; column 19, lines 15-32; and Figure 4) and the stored scheduling information associated with the certain instructions (Mahalingaiah column 17, line 66 to column 18, line 16; and Figure 4).

13. Referring to claim 4, Mahalingaiah has taught wherein the control logic is further adapted to cause the certain one of the instructions to be issued in accordance with a cycle within the loop iteration stage (Mahalingaiah column 17, line 66 to column 18, line 16; column 19, lines 15-32; and Figure 4) and the stored scheduling information associated with the certain instructions (Mahalingaiah column 17, line 66 to column 18, line 16; and Figure 4).

14. Referring to claims 8, 10, 12, and 14, Mahalingaiah has taught

a. Control logic coupled to the buffer (Applicant's claim 8) (Mahalingaiah column 17, line 66 to column 18, line 16; column 19, lines 15-32; and Figure 4);

b. The control logic including:

i. A loop iteration register for storing a loop iteration parameter (Applicant's claims 8, 10, 12, and 14) (Mahalingaiah column 19, line 52 to column 20, line 5; column 22, line 49 to column 23, line 6; Figure 5; and Figure 7), and

ii. A loop cycles register for storing a loop cycles parameter (Applicant's claims 8, 10, 12, and 14) (Mahalingaiah column 19, line 52 to column 20, line 5; column 22, line 49 to column 23, line 6; Figure 5; and Figure 7);

c. Wherein the control logic is adapted to cause the plurality of instructions to be issued to the functional unit from the buffer in accordance with the loop iteration parameter, the loop cycles parameter and the stored scheduling information (Applicant's claim 8) (Mahalingaiah column 17, line 66 to column 18, line 16; column 19, lines 15-32; and Figure 4); and

d. The control logic including an iteration initiation register for storing a loop iteration initiation parameter (Applicant's claims 8, 10, 12 and 14) (Applicant's claims 8, 10, 12, and 14) (Mahalingaiah column 19, line 52 to column 20, line 5; column 22, line 49 to column 23, line 6; Figure 5; and Figure 7).

15. Referring to claims 9, 11, and 13, Mahalingaiah has taught wherein the stored set of instructions consists of a kernel set of loop instructions, and wherein the control logic is operative so that the functional unit executes a number of loop iterations of the kernel set of loop instructions, a prologue set of loop instructions different than the stored kernel set of loop instructions, and an epilogue set of loop instructions different than the stored kernel set of loop instructions in accordance with the stored kernel set of loop instructions and received loop

parameters (Applicant's claims 9, 11, and 13) (Mahalingaiah column 17, line 66 to column 18, line 16; and Figure 4). In regards to Mahalingaiah, the MROM Access stores all of the MROM instructions, which includes all loop instructions that includes the kernel instructions.

16. Referring to claims 15 and 16, Mahalingaiah has taught a fetch unit, wherein the control logic is further operative so that the fetch unit can be shut down during execution of a number of iterations of the loop body corresponding to the stored plurality of instructions (Mahalingaiah column 18, lines 58-60).

17. Referring to claims 17 and 18, Mahalingaiah has taught a fetch unit, wherein the control logic is further operative so that the fetch unit can be shut down during execution of the prologue, kernel, and epilogue sets of loop instructions (Mahalingaiah column 18, lines 58-60).

18. Referring to claim 54, Mahalingaiah in view of George has taught wherein the buffer is further adapted to subsequently receive from the fetch stage and store a second different plurality of instructions, thereby overwriting the previously stored plurality of instructions, before issue to the functional unit and further adapted to store second different scheduling information associated with the second plurality of instructions, wherein the second plurality of the instructions comprise a set of instructions from a second different loop body, and wherein the second instructions are issued to the functional unit from the buffer in accordance with the stored second scheduling information so as to cause the functional unit to execute a second number of iterations of the second loop body (George Abstract; column 1, lines 26-47; and column 1, line 55 to column 2, line 68).

19. Claims 5-7 are rejected under 35 U.S.C. 103(a) as being unpatentable over Mahalingaiah, U.S. Patent Number 5,989,865 (herein referred to as Mahalingaiah) in view of Subramanian et al., U.S. Patent Number 5,867,711 (herein referred to as Subramanian).

20. Referring to claims 5-7, Mahalingaiah has not taught:

a. Wherein the scheduling information comprises a plurality of loop stage bit masks respectively associated with the plurality of instructions (Applicant's claim 5);

b. Wherein the scheduling information comprises a plurality of loop stage bit masks respectively associated with the plurality of instructions (Applicant's claims 6 and 7);

c. Wherein the control logic is further adapted to cause the certain one of the instructions to be issued in accordance with the respective one of the loop stage bit masks associated with the certain one of the instructions (Applicant's claims 6 and 7);

21. Subramanian has taught:

a. Wherein the buffer is further adapted to store a plurality of loop stage bit masks respectively associated with the plurality of instructions (Applicant's claim 5) (Subramanian column 2, lines 10-16; column 5, lines 29-45; column 10, lines 5-9; Figure 4; and Figure 5). In regards to Subramanian, there is a time-stamp, which functions similarly to the stage bit masks, assigned to each instruction to denote when an instruction is to be executed.

b. Wherein the buffer is further adapted to store a plurality of loop stage bit masks respectively associated with the plurality of instructions (Applicant's claims 6 and

7) (Subramanian column 2, lines 10-16; column 5, lines 29-45; column 10, lines 5-9; Figure 4; and Figure 5).

c. Wherein the control logic is further adapted to cause the certain one of the instructions to be issued in accordance with the respective one of the loop stage bit masks associated with the certain one of the instructions (Applicant's claims 6 and 7) (Subramanian column 2, lines 10-16; column 5, lines 29-45; column 10, lines 5-9; Figure 4; and Figure 5). In regards to Subramanian, there is a time-stamp, which functions similarly to the stage bit masks, assigned to each instruction to denote when an instruction is to be executed.

22. In regards to Subramanian, the elements of the claims have been taught by Subramanian as necessary parts of modulo scheduling. A person of ordinary skill in the art at the time the invention was made would have recognized that modulo scheduling is a form of software pipelining specifically for loops, so that successive execution iterations are overlapped (Subramanian column 2, lines 8-10), thereby increasing processor efficiency and speed by executing multiple instructions at the same time. Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to incorporate the modulo scheduling of Subramanian in the device of Mahalingaiah to increase processor efficiency and speed.

23. Claims 26-31, 35-41, and 43-49 are rejected under 35 U.S.C. 103(a) as being unpatentable over Mahalingaiah, U.S. Patent Number 5,989,865 (herein referred to as Mahalingaiah) in view of George, U.S. Patent Number 4,626,988 (herein referred to as George)

and in further view of Subramanian et al., U.S. Patent Number 5,867,711 (herein referred to as Subramanian).

24. Referring to claim 26, Mahalingaiah has taught a buffer in the dispatch stage of a processor, the buffer being associated with a functional unit that executes instructions issued to it from the dispatch stage (Mahalingaiah column 4, lines 32-43; column 11, lines 13-37; Figure 1; column 17, line 66 to column 18, line 16; and Figure 4). Mahalingaiah has not taught a first portion for receiving from a fetch stage and storing a kernel set of loop instructions. George has taught a first portion for receiving from a fetch stage and storing a kernel set of loop instructions (George Abstract; column 1, lines 26-47; and column 1, line 55 to column 2, line 68). A person of ordinary skill in the art at the time the invention was made, and as taught by George, would have recognized that the loop buffer eliminates redundancy, reduces storage contention, and providing efficient management for loop mode operations (George column 1, lines 26-47 and column 2, lines 57-68). Therefore, it would have been obvious to a person of ordinary skill in the at the time the invention was made to incorporate the loop buffer of George in the device of Mahalingaiah to eliminate redundancy, reduce storage contention, and provide efficient management for loop mode operations. In addition, Mahalingaiah has not taught a plurality of modulo schedule stage identifiers respectively associated with the kernel set of loop instructions, wherein the instructions are issued to the functional unit from the buffer in accordance with the stored modulo schedule stage identifiers so as to cause the functional unit to execute a number of iterations of a loop. Subramanian has taught a plurality of modulo schedule stage identifiers respectively associated with the kernel set of loop instructions, wherein the instructions are issued to the functional unit from the buffer in accordance with the stored modulo schedule stage

identifiers so as to cause the functional unit to execute a number of iterations of a loop (Subramanian column 2, lines 10-16; column 5, lines 29-45; column 10, lines 5-9; Figure 4; and Figure 5). In regards to Subramanian, the elements of the claim have been taught by Subramanian as necessary parts of modulo scheduling. A person of ordinary skill in the art at the time the invention was made would have recognized that modulo scheduling is a form of software pipelining specifically for loops, so that successive execution iterations are overlapped (Subramanian column 2, lines 8-10), thereby increasing processor efficiency and speed by executing multiple instructions at the same time. Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to incorporate the modulo scheduling of Subramanian in the device of Mahalingaiah to increase processor efficiency and speed.

25. Referring to claims 27-31, Mahalingaiah has taught

a. Wherein the processor further includes control logic, the buffer being coupled to the control logic and the functional unit for causing the kernel set of loop instructions to be issued to the functional unit (Applicant's claim 27) (Mahalingaiah column 17, line 66 to column 18, line 16; column 19, lines 15-32; and Figure 4);

b. Wherein the loop instructions comprise undecoded instructions, and wherein the processor further includes a decode stage interposed between the functional unit and the buffer for decoding the instructions (Applicant's claim 30) (Mahalingaiah column 4, lines 32-44; column 9, lines 6-16; Figure 1; column 18, lines 50-57; and Figure 4); and

c. Wherein the loop instructions comprise decoded instructions in the form of functional unit control signals (Applicant's claim 31) (Mahalingaiah column 4, lines 32-44; column 9, lines 6-16; and Figure 1). In regards to Mahalingaiah, instructions are functional unit control signals since they control how the functional unit operates.

26. Mahalingaiah has not taught

a. The kernel set of loop instructions and modulo schedule stage identifiers (Applicant's claim 27);

b. Wherein the modulo schedule stage identifiers comprise bit fields (Applicant's claim 28);

c. The control logic determining whether to issue a certain one of the loop instructions to the functional unit in accordance with a bit position of a set bit in the bit field corresponding to the certain loop instruction (Applicant's claim 28); and

d. Wherein the control logic is operative to cause a number of loop iterations of the kernel set of loop instructions, a prologue set of loop instructions different than the kernel set of loop instructions, and an epilogue set of loop instructions different than the kernel set of loop instructions in accordance with the stored modulo schedule stage identifiers (Applicant's claim 29).

27. Subramanian has taught:

a. The kernel set of loop instructions and modulo schedule stage identifiers (Applicant's claim 27) (Subramanian column 2, lines 10-16; column 5, lines 29-45; column 10, lines 5-9; Figure 4; and Figure 5);

b. Wherein the modulo schedule stage identifiers comprise bit fields (Applicant's claim 28) (Subramanian column 3, lines 36-44);

c. The control logic determining whether to issue a certain one of the loop instructions to the functional unit in accordance with a bit position of a set bit in the bit field corresponding to the certain loop instruction (Applicant's claim 28) (Subramanian column 2, lines 10-16; column 4, lines 16-26; column 5, lines 29-45; column 10, lines 5-9; Figure 4; and Figure 5); and

d. Wherein the control logic is operative to cause a number of loop iterations of the kernel set of loop instructions, a prologue set of loop instructions different than the kernel set of loop instructions, and an epilogue set of loop instructions different than the kernel set of loop instructions in accordance with the stored modulo schedule stage identifiers (Applicant's claim 29) (Subramanian column 2, lines 10-16; column 5, lines 29-45; column 6, lines 4-19; column 10, lines 5-9; Figure 4; and Figure 5).

28. In regards to Subramanian, the elements of the claims have been taught by Subramanian as necessary parts of modulo scheduling. A person of ordinary skill in the art at the time the invention was made would have recognized that modulo scheduling is a form of software pipelining specifically for loops, so that successive execution iterations are overlapped (Subramanian column 2, lines 8-10), thereby increasing processor efficiency and speed by

executing multiple instructions at the same time. Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to incorporate the modulo scheduling of Subramanian in the device of Mahalingaiah to increase processor efficiency and speed.

29. Referring to claims 35 and 43, Mahalingaiah has taught a method for executing a number of iterations of a loop in a processor, the method comprising:

a. Storing the kernel set of loop instructions at a dispatch stage of the processor (Mahalingaiah column 17, line 66 to column 18, line 16; and Figure 4); and

b. Storing loop parameters in control logic associated with the stored loop instructions (Mahalingaiah column 17, line 66 to column 18, line 16; and Figure 4).

30. Mahalingaiah has not taught receiving the kernel set of loop instructions and loop parameters from an instruction stream fetched by the processor. George has taught receiving the kernel set of loop instructions and loop parameters from an instruction stream fetched by the processor (George Abstract; column 1, lines 26-47; and column 1, line 55 to column 2, line 68). A person of ordinary skill in the art at the time the invention was made, and as taught by George, would have recognized that the loop buffer eliminates redundancy, reduces storage contention, and providing efficient management for loop mode operations (George column 1, lines 26-47 and column 2, lines 57-68). Therefore, it would have been obvious to a person of ordinary skill in the at the time the invention was made to incorporate the loop buffer of George in the device of Mahalingaiah to eliminate redundancy, reduce storage contention, and provide efficient management for loop mode operations.

31. In addition, Mahalingaiah has not taught:

a. The loop including a prologue set of loop instructions, a kernel set of loop instructions and an epilogue set of loop instructions; and

b. Causing the stored kernel set of instructions to be selectively issued to functional units of the processor in accordance with the stored loop parameters so that the functional units of the processor execute the number of iterations of the kernel set of loop instructions, the prologue set of loop instructions and the epilogue set of loop instructions based on the stored loop instructions.

32. Subramanian has taught:

a. The loop including a prologue set of loop instructions, a kernel set of loop instructions and an epilogue set of loop instructions (Subramanian column 6, lines 4-19 and Figure 5); and

b. Causing the stored kernel set of instructions to be selectively issued to functional units of the processor in accordance with the stored loop parameters so that the functional units of the processor execute the number of iterations of the kernel set of loop instructions, the prologue set of loop instructions and the epilogue set of loop instructions based on the stored loop instructions (Subramanian column 2, lines 10-16; column 5, lines 29-45; column 10, lines 5-9; Figure 4; and Figure 5).

33. In regards to Subramanian, the elements of the claim have been taught by Subramanian as necessary parts of modulo scheduling. A person of ordinary skill in the art at the time the invention was made would have recognized that modulo scheduling is a form of software pipelining specifically for loops, so that successive execution iterations are overlapped

(Subramanian column 2, lines 8-10), thereby increasing processor efficiency and speed by executing multiple instructions at the same time. Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to incorporate the modulo scheduling of Subramanian in the device of Mahalingaiah to increase processor efficiency and speed.

34. Referring to claims 36-40 and 44-48, Mahalingaiah has taught

a. A loop iteration register for storing a loop iteration parameter (Applicant's claims 36 and 44) (Mahalingaiah column 19, line 52 to column 20, line 5; column 22, line 49 to column 23, line 6; Figure 5; and Figure 7), and

b. A loop cycles register for storing a loop cycles parameter (Applicant's claims 36 and 44) (Mahalingaiah column 19, line 52 to column 20, line 5; column 22, line 49 to column 23, line 6; Figure 5; and Figure 7);

35. Mahalingaiah has not taught:

a. The control logic including an iteration initiation register for storing a loop iteration initiation parameter (Applicant's claims 36 and 44);

b. Adjusting the value in the loop cycles register from the stored loop cycles parameter in accordance with a processor cycle (Applicant's claims 37 and 45);

c. Resetting the value in the loop cycles register in accordance with the adjusted value and the stored iteration initiation parameter (Applicant's claims 37 and 45);

d. Adjusting the value in the loop iteration register in accordance with the resetting step (Applicant's claims 38 and 46);

e. Completing the issue of loop instructions in accordance with the adjusted value in the loop iteration register (Applicant's claims 38 and 46);

f. Storing a set of modulo schedule stage identifiers respectively associated with the kernel set of loop instructions, the step of causing the stored kernel set of loop instructions to be selectively issued including the step of comparing the stored loop parameters with the modulo schedule stage identifiers (Applicant's claims 39 and 47); and

g. Wherein the step of comparing the stored loop parameters with the modulo schedule stage identifiers includes the step of indexing to a certain one of the modulo schedule stage identifiers in accordance with a current cycle in the modulo schedule stage indicated by the stored loop parameters, the step of causing the stored kernel set of loop instructions to be selectively issued further including issuing the associated loop instruction to the functional unit if the certain modulo schedule stage identifier indicates that the functional unit is active (Applicant's claims 40 and 48).

36. Subramanian has taught:

a. The control logic including an iteration initiation register for storing a loop iteration initiation parameter (Applicant's claims 36 and 44) (Subramanian column 5, lines 49-56 and Figure 5);

b. Adjusting the value in the loop cycles register from the stored loop cycles parameter in accordance with a processor cycle (Applicant's claims 37 and 45) (Subramanian column 5, line 29 to column 7, line 7; Figure 4; and Figure 5);

c. Resetting the value in the loop cycles register in accordance with the adjusted value and the stored iteration initiation parameter (Applicant's claims 37 and 45) (Subramanian column 5, line 29 to column 7, line 7; Figure 4; and Figure 5);

d. Adjusting the value in the loop iteration register in accordance with the resetting step (Applicant's claims 38 and 46) (Subramanian column 5, line 29 to column 7, line 7; Figure 4; and Figure 5);

e. Completing the issue of loop instructions in accordance with the adjusted value in the loop iteration register (Applicant's claims 38 and 46) (Subramanian column 5, line 29 to column 7, line 7; Figure 4; and Figure 5);

f. Storing a set of modulo schedule stage identifiers respectively associated with the kernel set of loop instructions, the step of causing the stored kernel set of loop instructions to be selectively issued including the step of comparing the stored loop parameters with the modulo schedule stage identifiers (Applicant's claims 39 and 47) (Subramanian column 2, lines 10-16; column 5, lines 29-45; column 10, lines 5-9; Figure 4; and Figure 5); and

g. Wherein the step of comparing the stored loop parameters with the modulo schedule stage identifiers includes the step of indexing to a certain one of the modulo schedule stage identifiers in accordance with a current cycle in the modulo schedule stage indicated by the stored loop parameters, the step of causing the stored kernel set of loop instructions to be selectively issued further including issuing the associated loop instruction to the functional unit if the certain modulo schedule stage identifier indicates that the functional unit is active

(Applicant's claims 40 and 48) (Subramanian column 5, line 29 to column 7, line 7; Figure 4; and Figure 5).

37. In regards to Subramanian, the elements of the claims have been taught by Subramanian as necessary parts of modulo scheduling. A person of ordinary skill in the art at the time the invention was made would have recognized that modulo scheduling is a form of software pipelining specifically for loops, so that successive execution iterations are overlapped (Subramanian column 2, lines 8-10), thereby increasing processor efficiency and speed by executing multiple instructions at the same time. Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to incorporate the modulo scheduling of Subramanian in the device of Mahalingaiah to increase processor efficiency and speed.

38. Referring to claims 41 and 49, Mahalingaiah has taught shutting down the fetch unit during execution of the number of iterations of the loop (Mahalingaiah column 18, lines 58-60).

39. Claims 19-21 are rejected under 35 U.S.C. 103(a) as being unpatentable over Mahalingaiah as applied to claims 9, 11, and 13 above, in view of Valluri and Govindarajan's "Modulo-Variable Expansion Sensitive Scheduling" published in High Performance Computing, 1998 (herein referred to as Valluri). Mahalingaiah in view of Subramanian has not taught wherein the kernel set of loop instructions comprise MVE code. Valluri has taught wherein the kernel set of loop instructions comprise MVE code (Valluri section 1. Introduction, paragraphs 1-2). A person of ordinary skill in the art at the time the invention was made would have recognized that MVE is needed to handle overlapping of a single variable with a subsequent definition of itself by ensuring that different registers are used each time (Valluri section 1.

Introduction, paragraphs 1-2). Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to incorporate MVE of Valluri in the device of Mahalingaiah in view of Subramanian to handle same variable overlap.

40. Claims 22-25, 34, 42, and 50 are rejected under 35 U.S.C. 103(a) as being unpatentable over Mahalingaiah in view of Subramanian as applied to claims 3, 8, 11, and 13 above, and further in view of Mason et al., U.S. Patent Number 6,418,489 (herein referred to as Mason). Mahalingaiah has not taught wherein the control logic is further operative to allow interrupts to be handled at the end of a current loop iteration. Mason has taught wherein the control logic is further operative to allow interrupts to be handled at the end of a current loop iteration and to complete the number of loop iterations after the interrupt is handled (Mason column 6, lines 32-33). In regards to Mason, returning to complete the loop iterations is inherent to interrupts, since they are only temporary. Please see InstantWeb's Online Computing Dictionary. A person of ordinary skill in the art at the time the invention was made would have recognized that waiting until the end of a loop iteration to allow interrupts would ensure data is not lost and/or corrupted and continuing afterwards ensures the process completes and normal process has resumed. Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to incorporate the interrupts of Mason in Mahalingaiah.

41. Referring to claim 53, Mahalingaiah has taught a processor comprising:

a. A functional unit (Mahalingaiah column 4, lines 32-43; column 11, lines 13-37; and Figure 1);

b. A buffer that is coupled to provide instructions for execution by the functional unit (Mahalingaiah column 17, line 66 to column 18, line 16; and Figure 4);

c. Control logic coupled to the buffer (Mahalingaiah column 17, line 66 to column 18, line 16; column 19, lines 15-32; and Figure 4).

42. Mahalingaiah has not taught

a. Receive loop parameter associated with loop instructions issued to the functional unit from a fetch stage;

b. Cause a kernel set of the loop instructions issued to the functional unit to be stored in the buffer in accordance with the received loop parameters.

43. George has taught receiving

a. Receive loop parameter associated with loop instructions issued to the functional unit from a fetch stage (George Abstract; column 1, lines 26-47; and column 1, line 55 to column 2, line 68);

b. Cause a kernel set of the loop instructions issued to the functional unit to be stored in the buffer in accordance with the received loop parameters (George Abstract; column 1, lines 26-47; and column 1, line 55 to column 2, line 68).

44. A person of ordinary skill in the art at the time the invention was made, and as taught by George, would have recognized that the loop buffer eliminates redundancy, reduces storage contention, and providing efficient management for loop mode operations (George column 1, lines 26-47 and column 2, lines 57-68). Therefore, it would have been obvious to a person of ordinary skill in the at the time the invention was made to incorporate the loop buffer of George in the device of Mahalingaiah to eliminate redundancy, reduce storage contention, and provide efficient management for loop mode operations.

45. In addition, Mahalingaiah has not taught

a. Cause the functional unit to execute a number of iterations of the kernel set of the loop instructions based on the received loop parameter;

b. Cause the functional unit to further execute a prologue set of the loop instructions different from the stored kernel set of instructions based on the stored kernel set of instructions and the receive loop parameters; and

c. Cause the functional unit to further execute an epilogue set of the loop instructions different from the stored kernel set of instructions based on the stored kernel set of instructions and the received loop parameters.

46. Subramanian has taught

a. Cause the functional unit to execute a number of iterations of the kernel set of the loop instructions based on the received loop parameter (Subramanian column 2, lines 10-16; column 5, lines 29-45; column 10, lines 5-9; Figure 4; and Figure 5);

b. Cause the functional unit to further execute a prologue set of the loop instructions different from the stored kernel set of instructions based on the stored kernel set of instructions and the receive loop parameters (Subramanian column 6, lines 4-19 and Figure 5); and

c. Cause the functional unit to further execute an epilogue set of the loop instructions different from the stored kernel set of instructions based on the stored kernel set of instructions and the received loop parameters (Subramanian column 2, lines 10-16; column 5, lines 29-45; column 10, lines 5-9; Figure 4; and Figure 5).

47. In regards to Subramanian, the elements of the claim have been taught by Subramanian as necessary parts of modulo scheduling. A person of ordinary skill in the art at the time the invention was made would have recognized that modulo scheduling is a form of software pipelining specifically for loops, so that successive execution iterations are overlapped (Subramanian column 2, lines 8-10), thereby increasing processor efficiency and speed by executing multiple instructions at the same time. Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to incorporate the modulo scheduling of Subramanian in the device of Mahalingaiah to increase processor efficiency and speed.

Response to Arguments

48. Applicant's arguments with respect to claims 1-50 and 54 have been considered but are moot in view of the new ground(s) of rejection.

49. Applicant's arguments, see Amendment, filed 01 May 2006, with respect to the rejection(s) of claim(s) 53 have been fully considered and are persuasive. Therefore, the rejection has been withdrawn. However, upon further consideration, a new ground(s) of rejection is made in view of the above rejection.

Conclusion

50. The prior art made of record and not relied upon is considered pertinent to applicant's disclosure as follows. Applicant is reminded that in amending in response to a rejection of claims, the patentable novelty must be clearly shown in view of the state of the art disclosed by the references cited and the objections made. Applicant must also show how the amendments avoid such references and objections. See 37 CFR § 1.111(c).

a. Ganapathy et al., U.S. Patent Number 6,832,306, has taught a loop buffer that receives and stores loop instructions from a fetch stage.

51. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Aimee J. Li whose telephone number is (571) 272-4169. The examiner can normally be reached on M-T 7:30am-5:00pm.

52. If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Eddie Chan can be reached on (571) 272-4162. The fax phone number for the organization where this application or proceeding is assigned is 703-872-9306.

53. Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

AJL
Aimee J. Li
10 July 2006

EDDIE CHAN
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2100